Processus Unifié

Unified Software Development Process / Unified Process (UP)

Merci à tous ceux qui ont rendu leurs cours et exposés disponibles sur le web & dans les livres, voir Biblio. & refs dans les slides

M. Blay-Fornarino

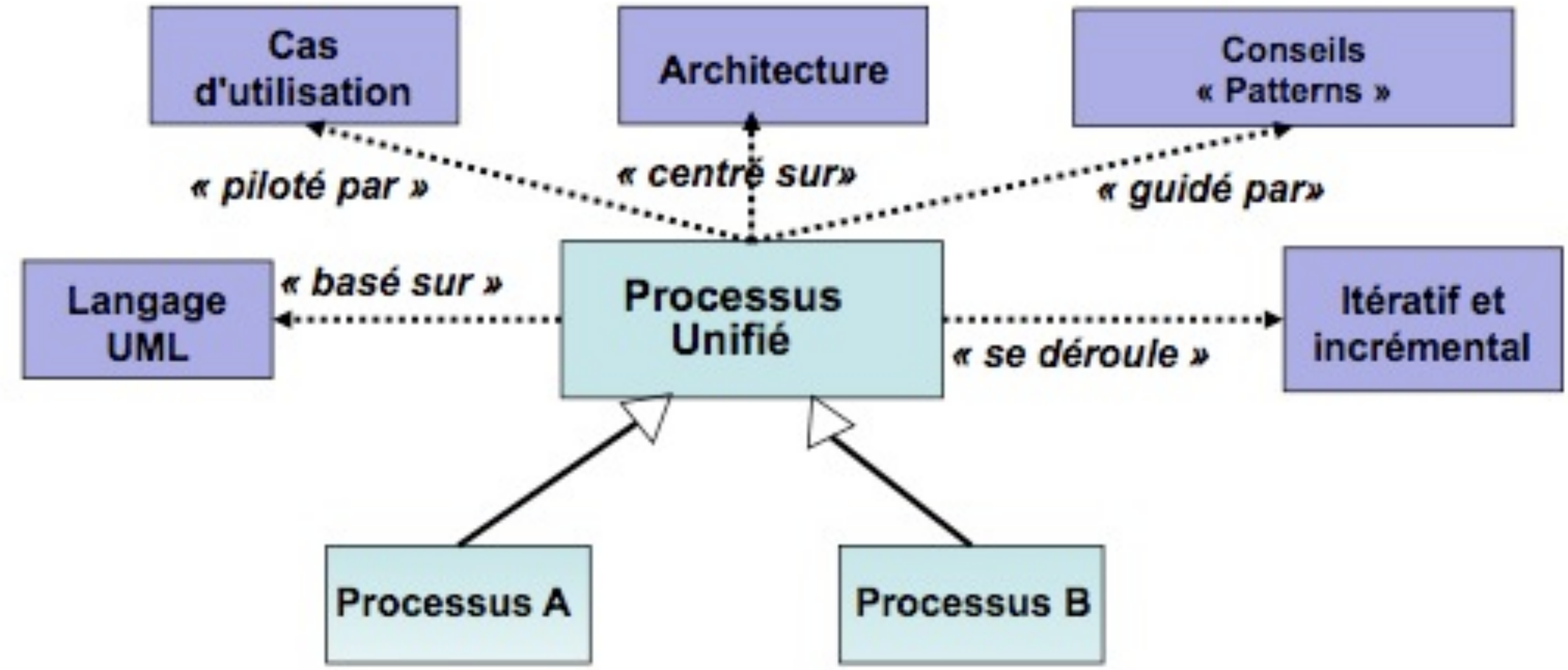
Unified Process (UP)

- Beaucoup de méthodes
  - liées à des outils, à l'adaptation à UML (comme langage de notation) de méthodes pré-existantes , aux entreprises, etc.
  - …finalement, autant de méthodes que de concepteurs / projets

➡ (1997) UP : Rumbaugh, Booch, Jacobson (les concepteurs d'UML)

- une trame commune des meilleures pratiques de developpement (pas une méthode)
- Issu des travaux de l'Object Management Group et de Rational
- Plusieurs variantes (RUP,UPEDU,…) toutes intégrant l'UML

Unified Software Development Process / Unified Process (UP)



Unified Software Development Process / Unified Process (UP)

Un Processus unifié est
un processus de développement logiciel
- construit sur UML
- conduit par les cas d'utilisation
- piloté par les risques
- centré sur l'architecture,
- itératif et incrémental

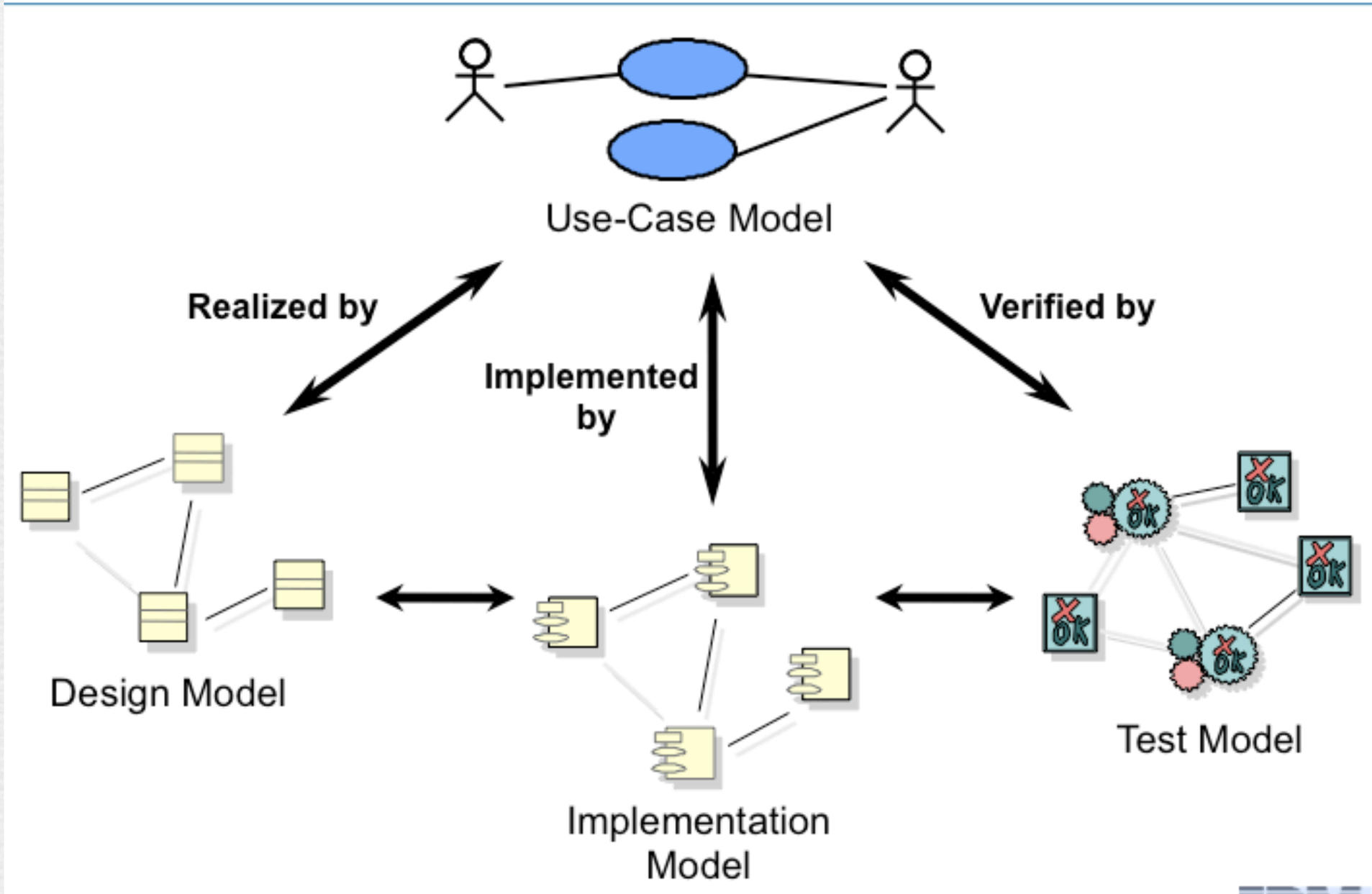
- organisé autour de 4 phases :
préétude (inception), élaboration, construction et transition

- défini par 6 disciplines fondamentales :
Modélisation métier, Analyse et Conception, Implémentation, Test et Déploiement

Processus Unifié
Piloté par les cas d'utilisation

Le processus de développement est centré sur l'utilisateur

Modèles UML



Unified Software Development Process / Unified Process (UP)

Un Processus unifié est
un processus de développement logiciel
- construit sur UML
- conduit par les cas d'utilisation
- piloté par les risques
- centré sur l'architecture,
- itératif et incrémental

- organisé autour de 4 phases :
préétude (inception), élaboration, construction et transition

- défini par 6 disciplines fondamentales :
Modélisation métier, Analyse et Conception, Implémentation, Test et Déploiement

(3) Pilotage par les risques

- **Un risque est un événement redouté dont l'occurrence est plus ou moins prévisible et provoquant, lorsqu'il se produit, des dommages sur le projet.**

- **Il ne faut pas confondre risque et problème.**

  Un problème est un risque qui s'est révélé.

(3) Pilotage par les risques

- **Un risque est un événement redouté dont l'occurrence est plus ou moins prévisible et provoquant, lorsqu'il se produit, des dommages sur le projet.**

- **Il ne faut pas confondre risque et problème.**

  Un problème est un risque qui s'est révélé.

Facteurs de Risque

Quelques facteurs de Risques :

- Technique/ Architectural
  - Technologie incertaine, visibilité partielle
- Ressources
  - Les gens, les compétences, le financement
- «Business»
  - La concurrence, les interfaces avec les fournisseurs
- Planning
  - Dépendances
  - «Only 24 hours in a day»
- Changements d'exigences

A prendre en compte dans le cahier des charges

Doivent être identifiés et «priorisés» dans des artefacts dédiés

An ongoing or upcoming concern that has a significant probability of adversely affecting the success of major milestones. (RUP Glossary)

Identifier les facteurs de risques

- Un facteur de risque peut entraîner l'apparition de plusieurs risques

  Développement sans spécifications approuvées

  -> insatisfaction du client

  -> surcoût du projet

- Plusieurs facteurs de risque peuvent contribuer à l'apparition d'un même risque

  Connaissance insuffisante du métier client,

  Plan de validation incomplet,

  Conduite au changement non-prévue

  -> rejet par les utilisateurs

Analyser les risques

- L'analyse des risques consiste à évaluer :
  - La probabilité d'apparition d'un risque
  - La gravité d'une conséquence

Exemple :

Probabilité de rencontrer un chat en liberté à Paris : forte
Probabilité de rencontrer un lion en liberté à Paris : faible
Probabilité de rencontrer un lion en liberté dans la savane : forte
Probabilité de rencontrer un chat en liberté dans la savane : faible

Gravité d'une morsure de chat : faible
Gravité d'une morsure de lion : forte

Actions

- La gestion des risques consiste à prévoir ou à anticiper les situations risquées et mettre en œuvre un plan d'actions composé :
  - **Actions de réduction** : réduire l'influence des facteurs de risques (réduire la probabilité)
    - Les technologies utilisées sont bien connues des développeurs, ce qui devrait réduire les risques de non-aboutissement.
  - **Actions préventives** : ne pas se mettre dans une situation (actions sur le déclenchement des facteurs de risques)
    - L'intervention d'un ergonome réduira les risques de non acceptance.
  - **Actions de couverture** : limiter les conséquences des risques (réduire la gravité)
    - Un développement itératif, incrémental permettra de limiter la prise de risque technologique
  - ***Attitude de NO GO***
    - Le projet est abandonné.

Unified Software Development Process / Unified Process (UP)

Un Processus unifié est
un processus de développement logiciel
- construit sur UML
- conduit par les cas d'utilisation
- piloté par les risques
- centré sur l'architecture,
- itératif et incrémental

- organisé autour de 4 phases :
préétude (inception), élaboration, construction et transition

- défini par 6 disciplines fondamentales :
Modélisation métier, Analyse et Conception, Implémentation, Test et Déploiement

Notion d'architecture

- Au sens de RUP, une architecture :
  - Sert à comprendre le système lorsqu'il est complexe
  - Pilote le projet en découpant les tâches
  - Favorise la réutilisation

> Software architecture is not only concerned with structure and behavior, but also with usage, functionality, performance, resilience, reuse, comprehensibility, economic and technological constraints and tradeoffs, and esthetics. (RUP, 98)-

> A Technical Architecture is the minimal set of rules governing the arrangement, interaction, and interdependance of the parts or elements that together may be used to form an information system. (U.S. Army 1996)

Quelques axes pour considérer l'architecture

- On pourra parler
  - d'architecture logicielle (ou architecture logique) : organisation à grande échelle des classes logicielles en packages, sous-systèmes et couches
  - d'architecture de déploiement : décision de déploiement des différents éléments
  - Notion de patterns architecturaux
    - ex. : Couches, MVC…

Quelques axes pour considérer l'architecture

- Architectures client/serveurs en niveaux
  - aussi appelés tiers
- Architectures en couches
  - Présentation, Application, Domaine/métier, Infrastructure métier (services métiers de bas-niveau), Services techniques (ex. sécurité), Fondation (ex. accès et stockage des données)
- Architectures en zones de déploiement
  - déploiement des fonctions sur les postes de travail des utilisateurs (entreprise : central/départemental/local)
- Architectures à base de composants
  - réutilisation de composants

Processus centré sur l'architecture

- Les cas d'utilisation ne sont pas suffisants comme lien pour l'ensemble des membres du projet
- L'architecture joue également ce rôle, en insistant sur la réalisation concrète de prototypes incrémentaux qui «démontrent» les décisions prises
- D'autre part
  - plus le projet avance, plus l'architecture est difficile à modifier
  - les risques liés à l'architecture sont très élevés, car très coûteux
- Objectifs pour le projet
  - établir dès la phase d'élaboration des fondations solides et évolutives pour le système à développer, en favorisant la réutilisation
  - l'architecture s'impose à tous, contrôle les développements ultérieurs, permet de comprendre le système et d'en gérer la complexité

Processus Unifié
Centré sur l'architecture

- L'architecture regroupe les différentes vues du système qui doit être construit.
  - Elle doit prévoir la réalisation de tous les cas d'utilisation.
- Marche à suivre:
  - Créer une ébauche grossière de l'architecture.
    - Travailler sur les cas d'utilisation représentant les fonctions essentielles.
    - Adapter l'architecture pour qu'elle prenne en compte ces cas d'utilisation.
  - Sélectionner d'autres cas d'utilisation et refaire de même.
- L'architecture et les cas d'utilisation évoluent de façon concomitante.

Unified Software Development Process / Unified Process (UP)

Un Processus unifié est
un processus de développement logiciel
- construit sur UML
- conduit par les cas d'utilisation
- piloté par les risques
- centré sur l'architecture,
- itératif et incrémental

- organisé autour de 4 phases :
préétude (inception), élaboration, construction et transition

- défini par 6 disciplines fondamentales :
Modélisation métier, Analyse et Conception, Implémentation, Test et Déploiement

Processus Unifié
Itératif et Incrémental

Figure 1. — Itératif / incrémental (Jeff Patton)

http://www.entreprise-agile.com/HistoAgile.pdf

Processus Unifié
Itératif et Incrémental

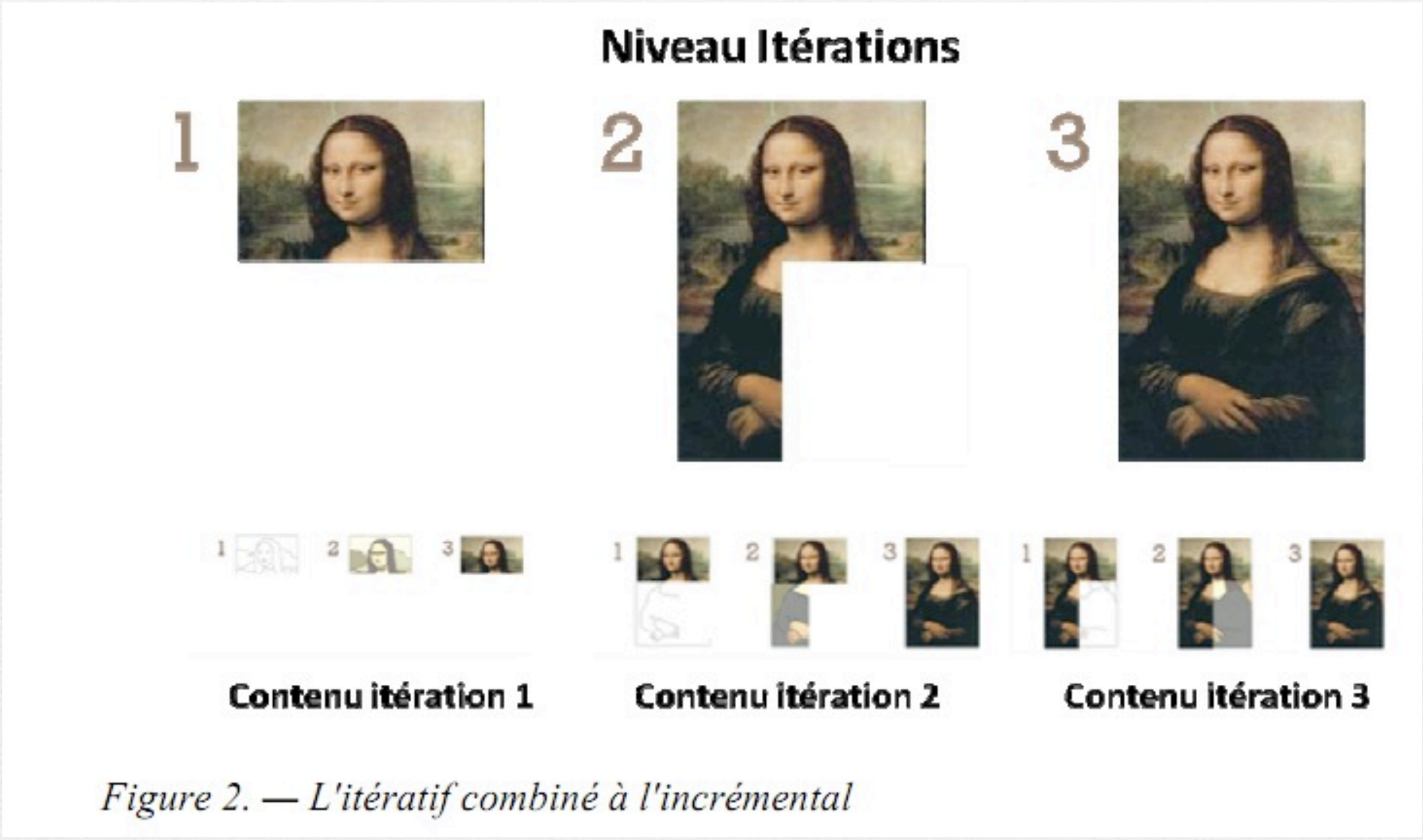


Figure 2. — L'itératif combiné à l'incrémental

http://www.entreprise-agile.com/HistoAgile.pdf

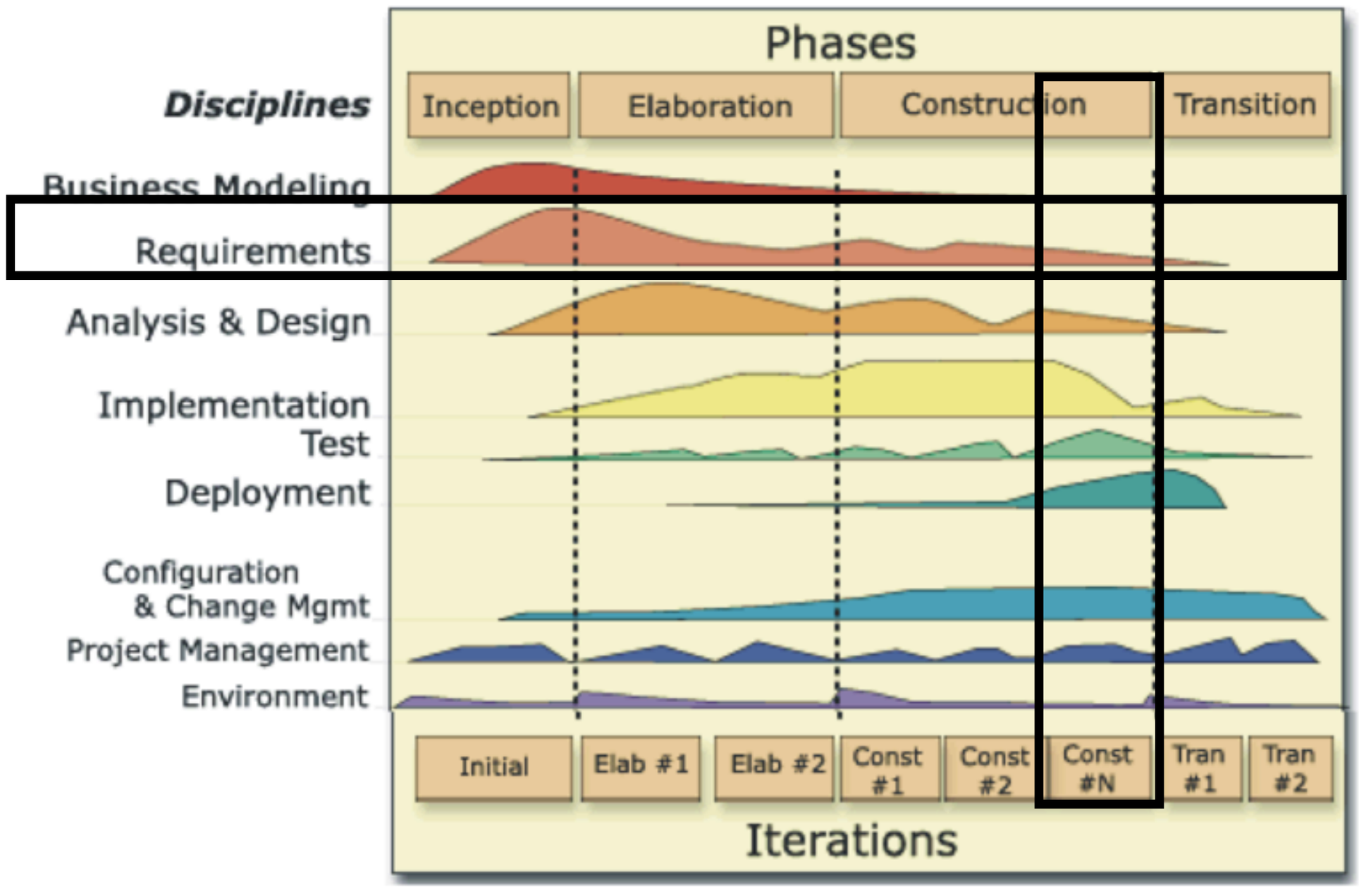
Processus Unifié
Itératif et Incrémental

- Ordonnancement des itérations basé sur les priorités entre cas d'utilisation et sur l'étude du risque
- Une itération est une séquence **d'activités**
- Une itération se décompose en:
  - Une planification de l'itération
  - Analyse des besoins (raffinement)
  - Analyse et conception
  - Implémentation et tests
  - Évaluation
  - Livraison

Approche Itérative

time

content

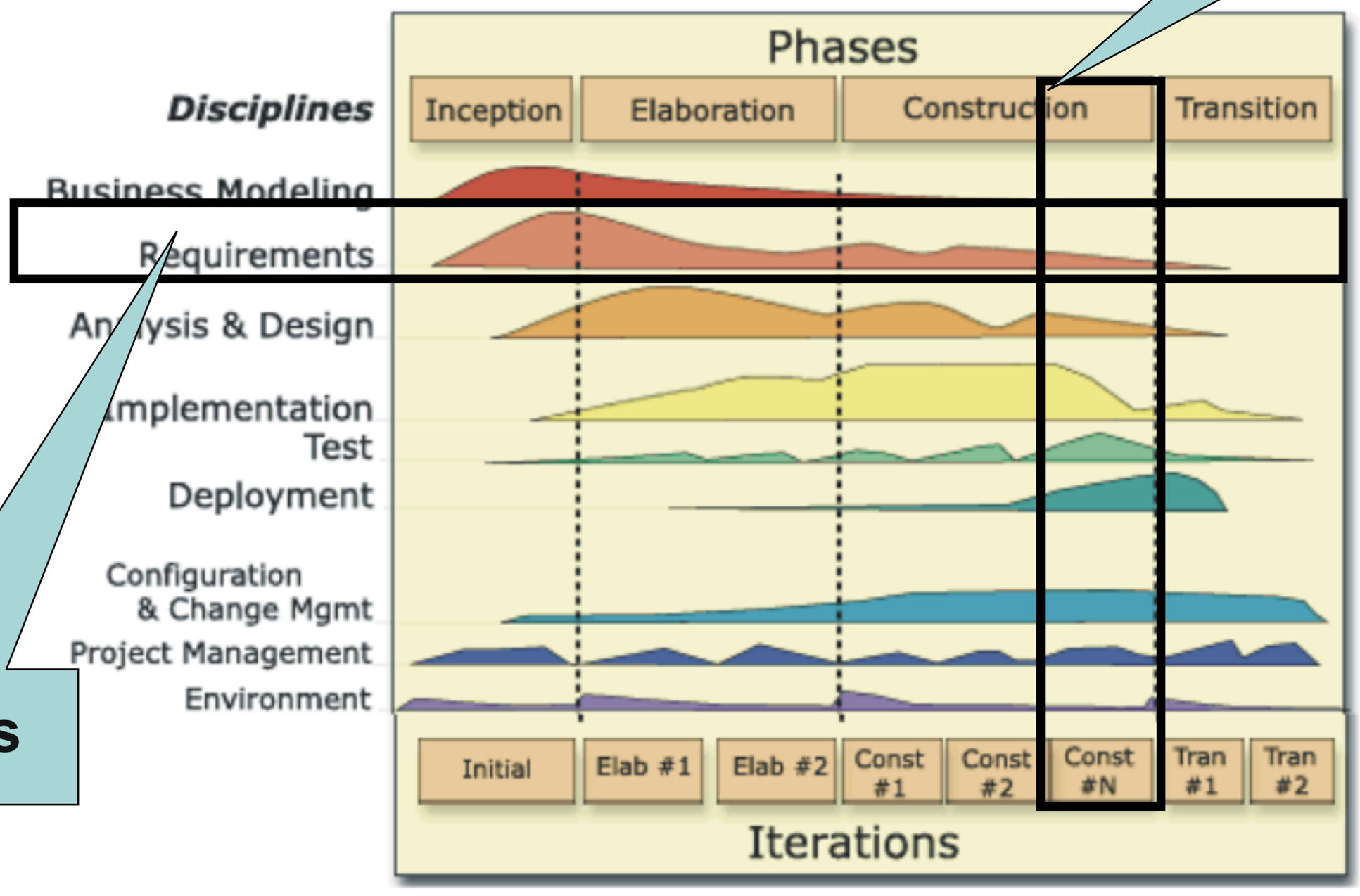


Approche Itérative

Une **itération**, traverse toutes les «disciplines.

time

content



Disciplines

Unified Software Development Process / Unified Process (UP)

Un Processus unifié est
un processus de développement logiciel
- construit sur UML
- conduit par les cas d'utilisation
- piloté par les risques
- centré sur l'architecture,
- itératif et incrémental

- organisé autour de 4 phases :
préétude (inception), élaboration, construction et transition

- défini par 6 disciplines fondamentales :
Modélisation métier, Analyse et Conception, Implémentation, Test et Déploiement

Suite

Pré-étude (Inception Phase): *Objectifs*

- Établir la portée du projet et les conditions aux limites
- Déterminer les cas d'utilisation et les scénarios principaux
- Proposer une architecture
- Estimer le coût global et le calendrier
- Identifier les risques potentiels et leurs sources
- Démontrer que le système proposé est en mesure de résoudre les problèmes ou de prendre en charge les objectifs fixés

Vision : Glossaire, Détermination des parties prenantes et des utilisateurs, Détermination de leurs besoins, Besoins fonctionnels et non fonctionnels, Contraintes de conception

Pré-étude (Inception Phase): *Evaluation*

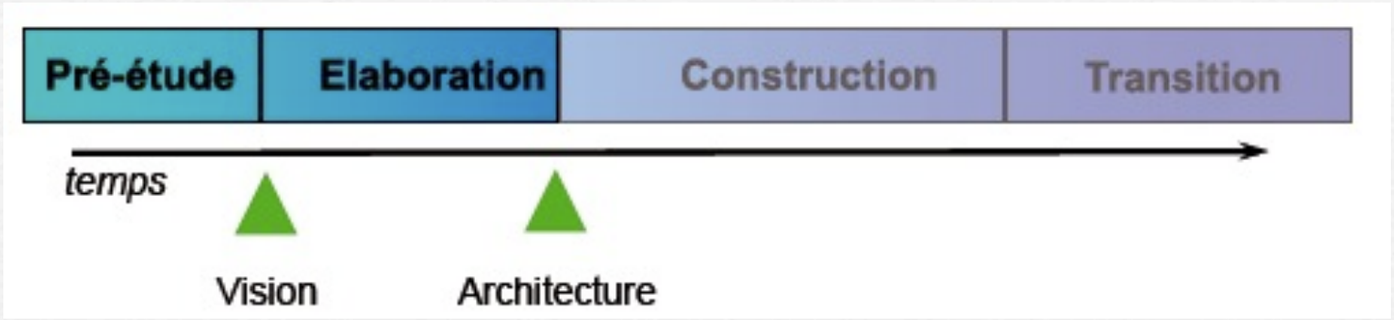
- Acceptation des parties prenantes sur la définition du champ d'application
- Accord sur les exigences (ensemble et compréhension commune)
- Accord sur les estimations de coût, le planning, les priorités, les risques et le processus de développement
- Les risques ont été identifiés et une stratégie de gestion du risque a été prévue.

Pré-étude (Inception Phase): *Evaluation*

- Acceptation des parties prenantes sur la définition du champ d'application
- Accord sur les exigences (ensemble et compréhension commune)
- Accord sur les estimations de coût, le planning, les priorités, les risques et le processus de développement
- Les risques ont été identifiés et une stratégie de gestion du risque a été prévue.

Abandon du projet si pas de passage de ce «jalon»

Elaboration Phase: *Objectifs*



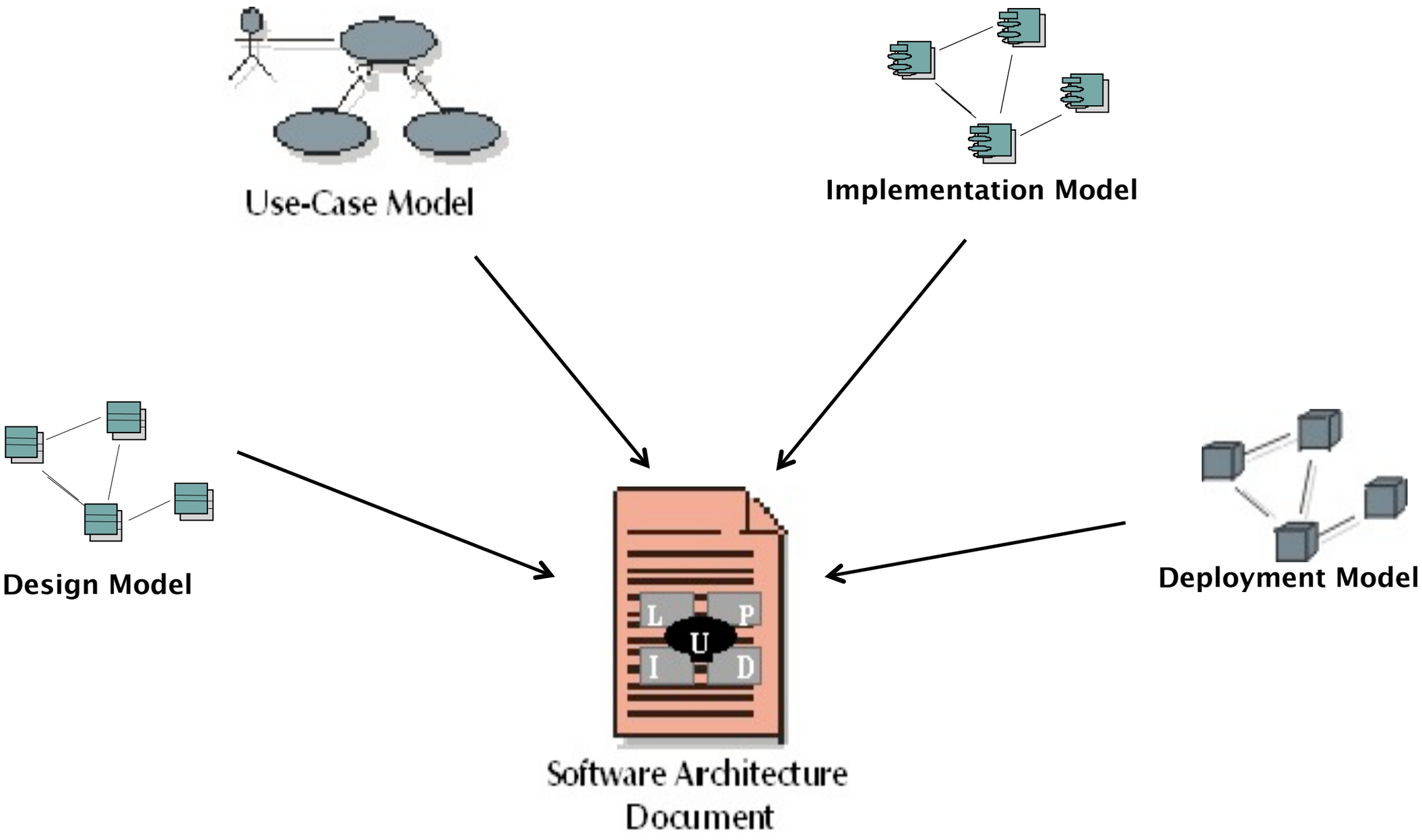
- Définir et valider l'architecture de base
- Formuler les cas d'utilisation pour couvrir environ 80% des besoins fonctionnels
- Définir les niveaux de qualité à atteindre,
- Etablir un plan détaillé pour la phase de construction
- Démontrer que l'architecture de base prend en charge «la vision» à un coût raisonnable dans un délai raisonnable

Architecture : Document d'architecture Logicielle,
Différentes vues selon la partie prenante,
Comportement et conception des composants du système

Description d'Architecture (1)

- «Software Architecture Document»
  - Donne une vue d'ensemble complète de l'architecture du système logiciel
  - Inclus
    - Vues d'Architecture
    - Buts et contraintes
      - Exigences que l'architecture doit supporter
      - Contraintes Techniques
    - Caractéristiques de taille et performance
    - Qualité, extensibilité, and cibles potentielles (portabilité)

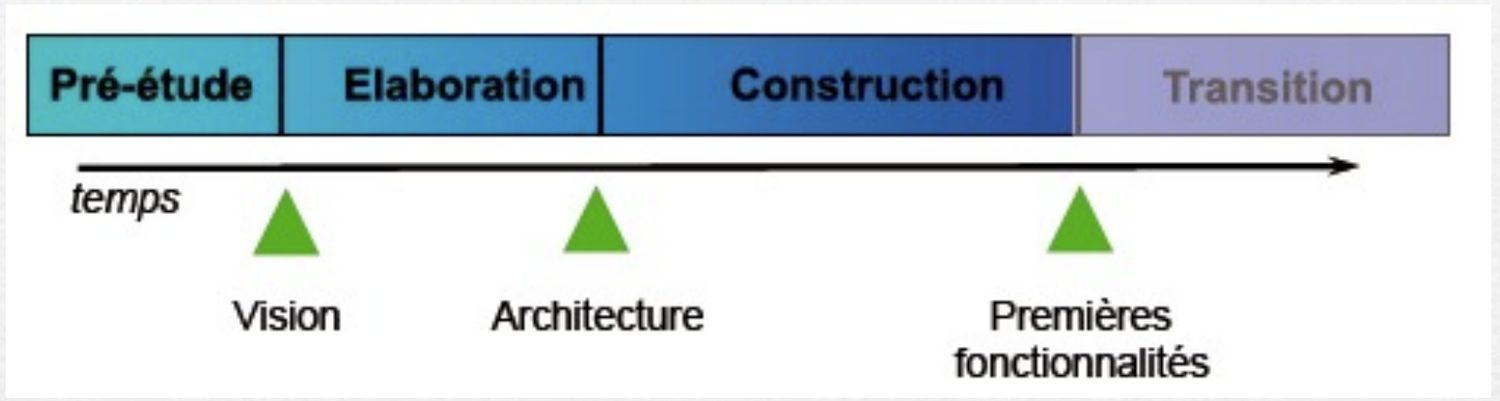
Description d'Architecture (2)



Elaboration Phase: *Evaluation*

- Vision du produit et les exigences sont stables.
- L'architecture est stable.
- Les éléments de risque majeurs ont été abordés et résolus.
- Les plans d'itération pour la phase de construction sont suffisamment détaillés
- Tous les intervenants s'entendent pour dire que la vision actuelle peut être atteinte si le plan actuel est exécuté dans le contexte de l'architecture actuelle.

Construction Phase: *Objectifs*



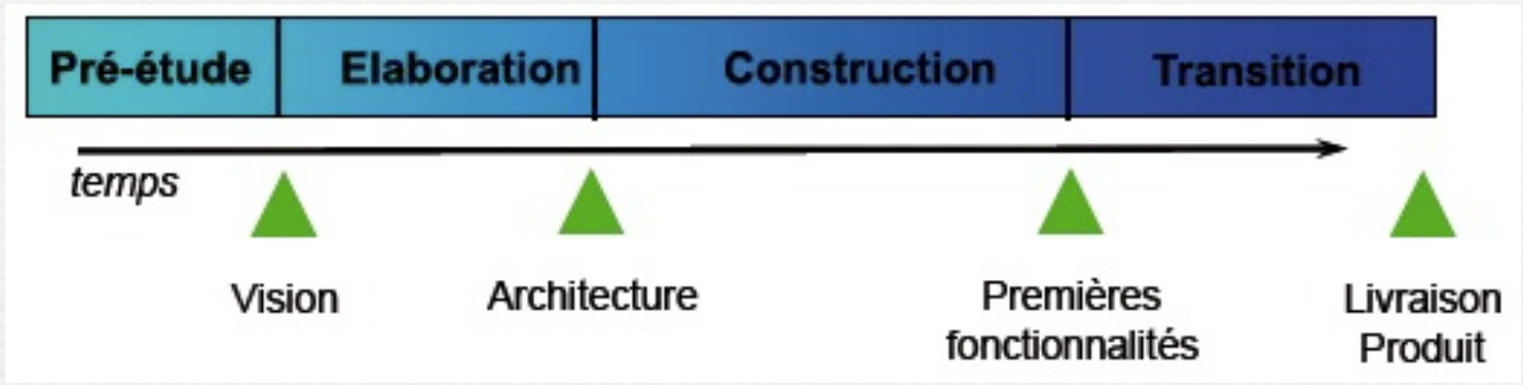
- Objectifs principaux
  - Produire un logiciel utilisable conforme aux besoins
  - Confronter ce logiciel aux critères d'acceptation (élaborés pendant la phase d'opportunité)
- Extension de l'identification, de la description et de la réalisation des cas d'utilisation
- Finalisation de l'analyse, de la conception, de l'implémentation et des tests

Produit : Première version suffisamment stable et mature pour être déployée (testée, documentée, accompagnée…), Les procédures d'installation sont prêtes

Construction Phase: *Evaluation*

- La version du produit est-elle assez stable et mûre pour être déployée dans la communauté des utilisateurs?
- Les utilisateurs sont-ils informés ? Sensibilisés ? Formés?
- Les dépenses engagées par rapport aux prévisions sont-elles acceptables?

Transition Phase: *Objectifs*



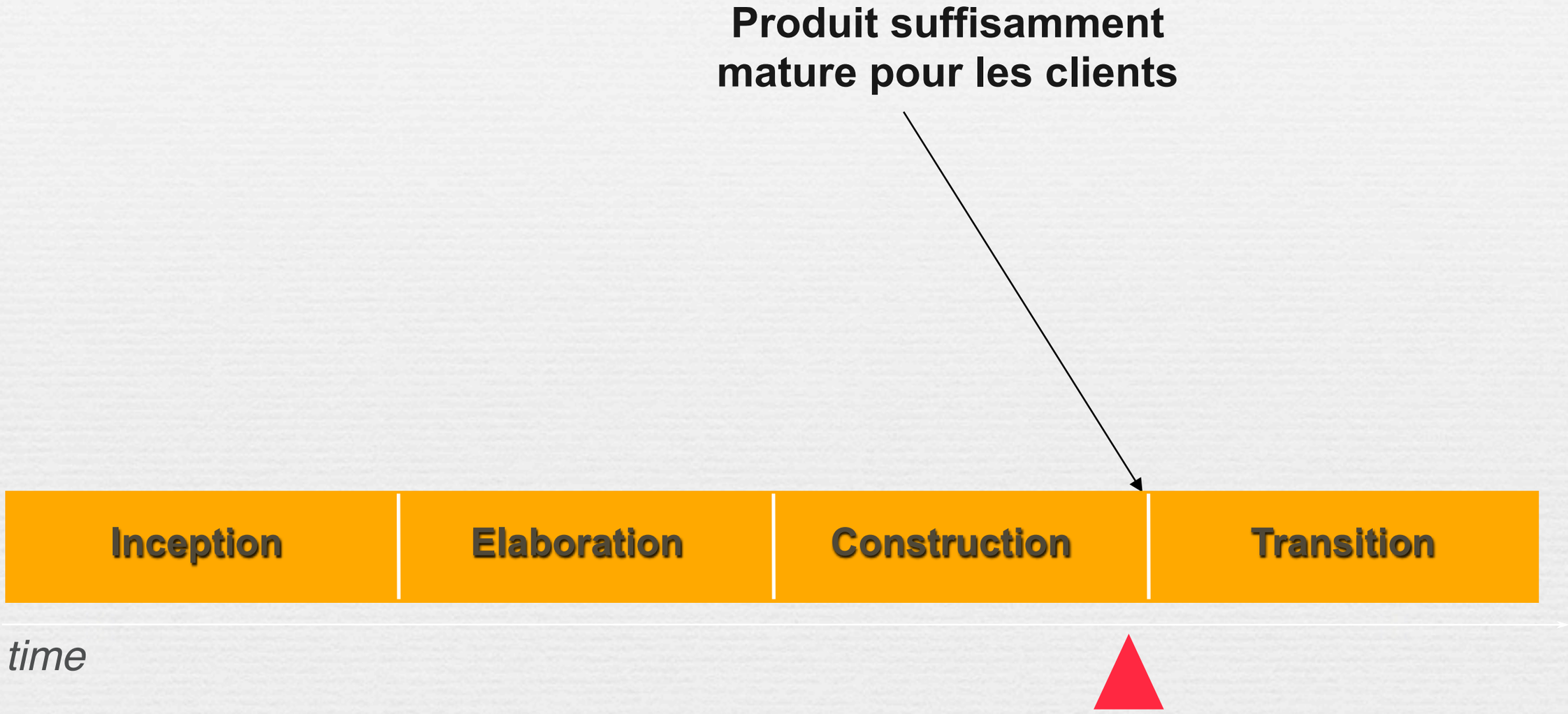
- Préparation des activités
- Recommandations au client sur la mise à jour de l'environnement logiciel
- Elaboration des manuels et de la documentation concernant la version du produit
- Adaptation du logiciel
- Correction des anomalies liées au béta test
- Dernières corrections

Livraison du produit aux utilisateurs

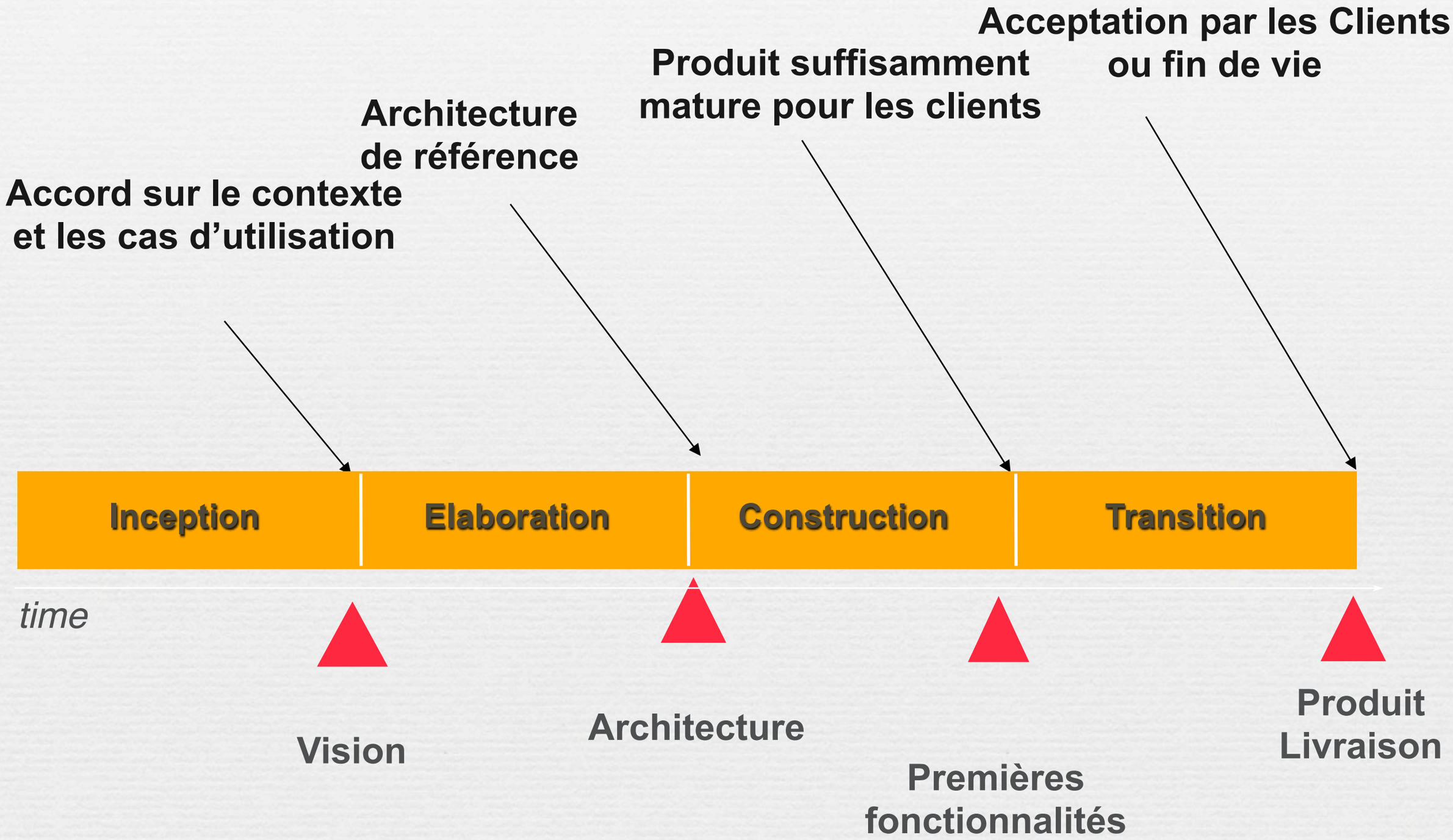
Transition Phase: *Evaluation*

- Les utilisateurs sont-ils satisfaits ?
- Les dépenses engagées par rapport aux prévisions sont-elles acceptables?

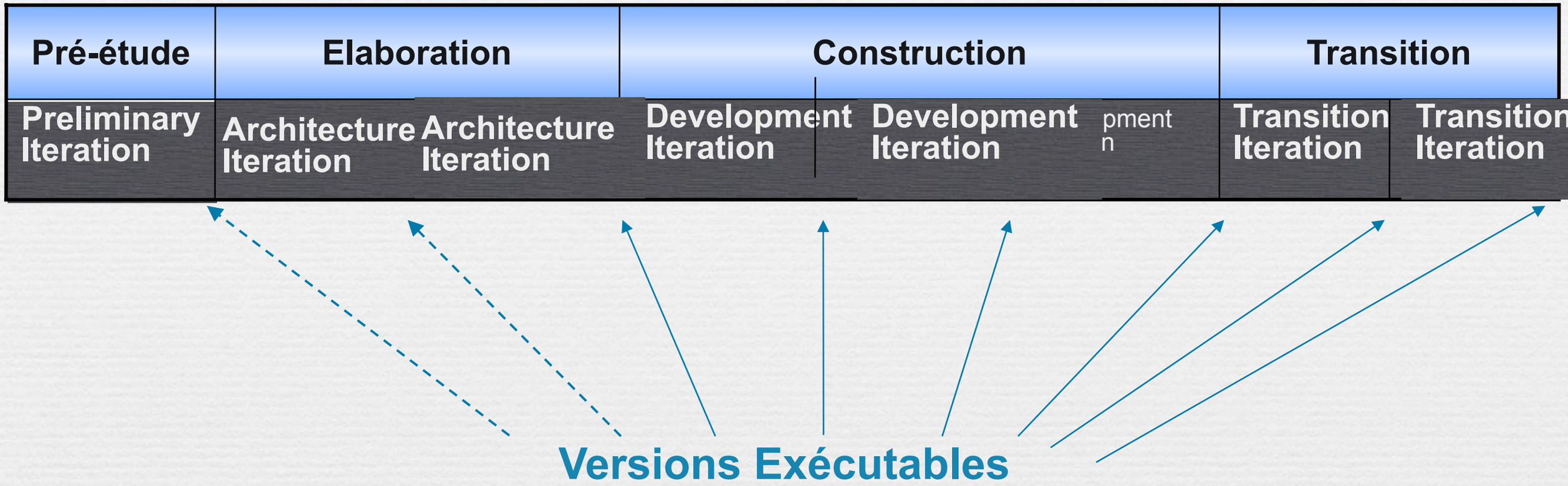
Principaux jalons



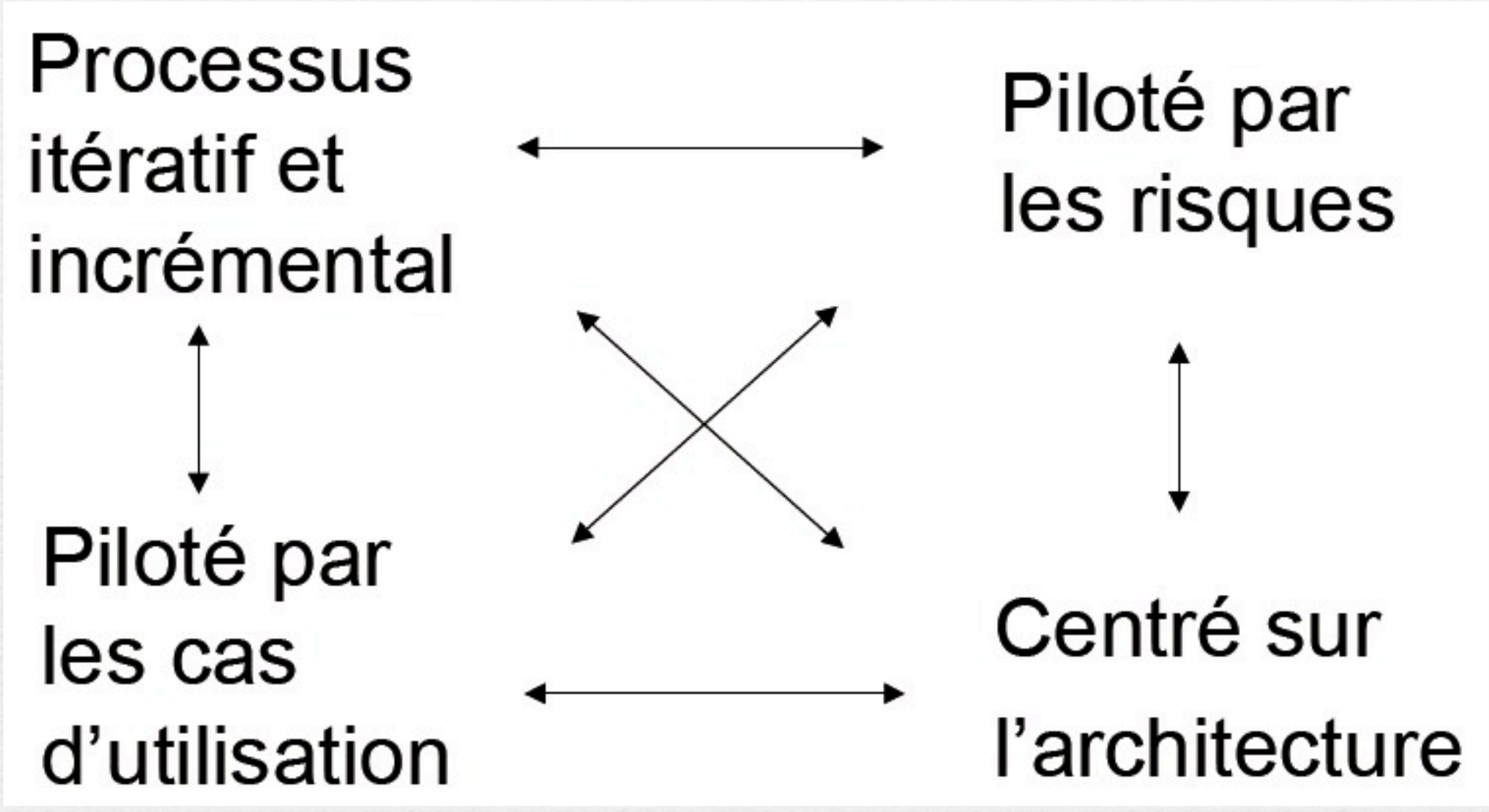
Principaux jalons



Itérations dans chaque Phase



Tous les critères caractérisant les processus UP sont liés



Variantes UP

- Il existe plusieurs variantes:
  - RUP: Rational Unified Process (IBM)
    - Version industrielle avec une panoplie d'outils
  - UPEDU:
    - Version allégée pour des environnements académiques
- Toutes ces variantes:
  - élaborent des modèles qui seront intereliés
  - ont des mécanismes d'évaluation et d'adaptation du processus

Un exemple de bout en bout

Basé sur le cours de
M1 MIAGE - SIMA 2005-2006 / Yannick Prié -
Université Claude Bernard Lyon 1

Pour mieux comprendre le procédé
mais pas à savoir faire

Exemple : liste initiale des besoins

- Une chaîne d'hôtels a décidé de mettre sur le réseau un système de réservation de chambres ouvert à tout client via un navigateur, et d'autre part elle veut automatiser la gestion de ses hôtels. Un hôtel est géré par un directeur assisté d'employés.
- Pour réserver à distance, après avoir choisi hôtels et dates, le client fournit un numéro de carte bleue. Lorsque le retrait a été accepté (1h après environ), la réservation devient effective et une confirmation est envoyée par mail. Les clients sous contrat (agences de voyage...) bénéficient d'une réservation immédiate.
- Le directeur de l'hôtel enregistre les réservations par téléphone. Si un acompte est reçu avant 72h, la réservation devient effective, sinon elle est transformée en option (toute personne ayant payé a la priorité). Si la réservation intervient moins de 72h avant la date d'occupation souhaitée, le client doit se présenter avant 18h.
- Le directeur fait les notes des clients, perçoit l'argent et met à jour le planning d'occupation effectif des chambres.
- Une chambre est nettoyée soit avec l'accord du client lorsqu'il reste plusieurs jours, soit après le départ du client s'il s'en va, et dans ce cas avant occupation par un nouveau client. Les employés s'informent des chambres à nettoyer et indiquent les chambres nettoyées au fur et à mesure. Pour cela les chambres vides à nettoyer doivent être affichées, et les employés doivent pouvoir indiquer les chambres nettoyées de façon très simple. Un historique des chambres nettoyées par chaque employé est conservé un mois.

Exemple : liste initiale des besoins

- Une chaîne d'hôtels a décidé de mettre sur le réseau un système de réservation de chambres ouvert à tout client via un navigateur, et d'autre part elle veut automatiser la gestion de ses hôtels. Un hôtel est géré par un directeur assisté d'employés.
- ....
- Une chambre est nettoyée soit avec l'accord du client lorsqu'il reste plusieurs jours, soit après le départ du client s'il s'en va, et dans ce cas avant occupation par un nouveau client. Les employés s'informent des chambres à nettoyer et indiquent les chambres nettoyées au fur et à mesure. Pour cela les chambres vides à nettoyer doivent être affichées, et les employés doivent pouvoir indiquer les chambres nettoyées de façon très simple. Un historique des chambres nettoyées par chaque employé est conservé un mois.

Etude préliminaire

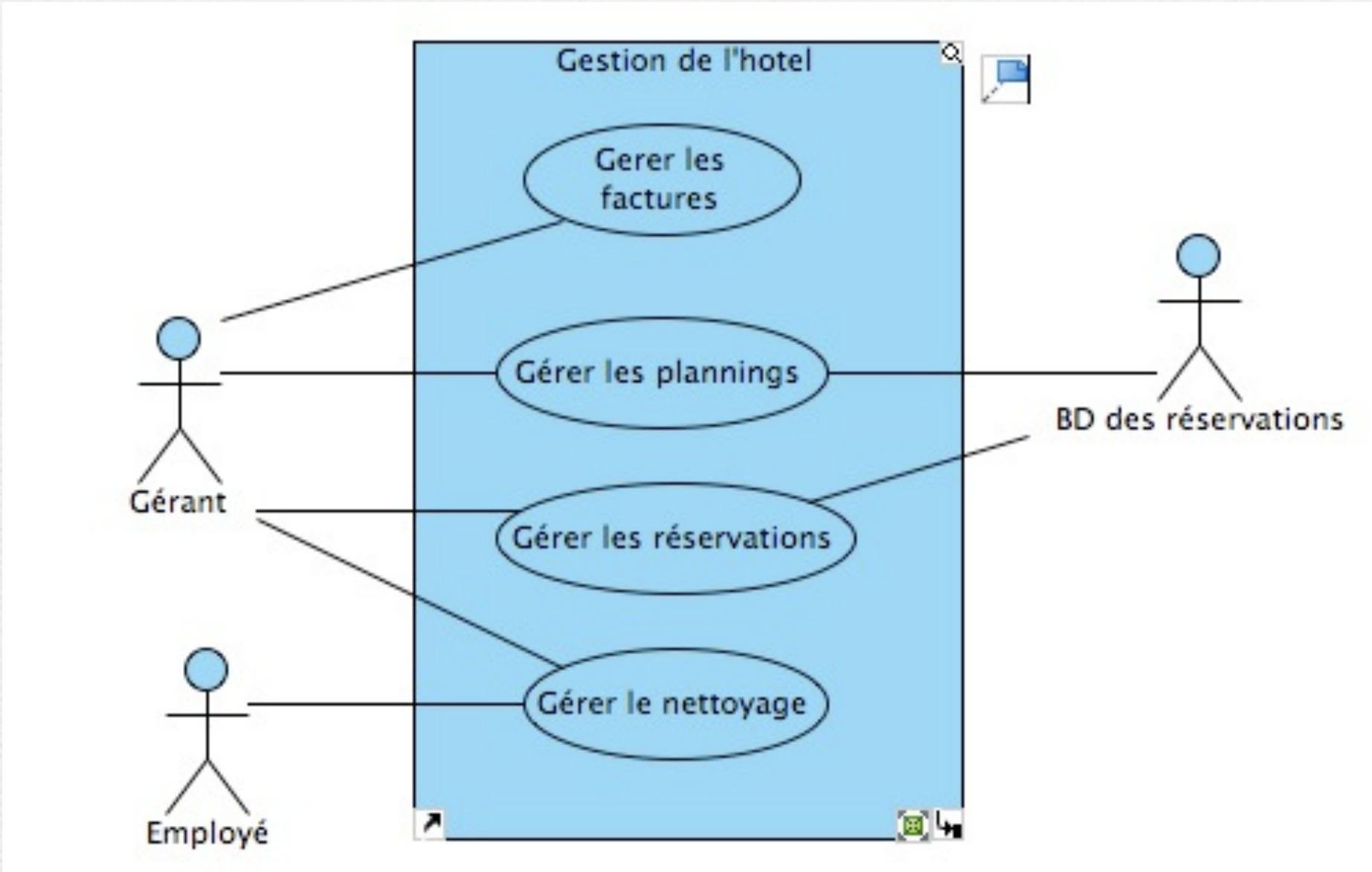
- Appréhender les besoins fonctionnels
  - trouver les acteurs
    - à partir des besoins
    - délimiter le système par rapport à son environnement (système = boîte noire)
    - chercher qui interagit avec le système (rôles)
  - trouver les cas d'utilisation
    - examiner comment chaque acteur interagit avec le système pour que celui-ci lui rende un service
    - regrouper les interactions similaires en cas d'utilisation
  - décrire brièvement les cas d'utilisation
  - construire le modèle des cas d'utilisation
    - les considérer dans leur ensemble

Ex : Etude préliminaire

Le système à développer est divisé en deux sous systèmes indépendants :

- le système de réservation à distance et le système de gestion local à l'hôtel.

La base de données des réservations est considérée comme un système externe.



Etude préliminaire

- Classer les cas d'utilisation par priorité
  - la priorité dépend des risques associés au cas d'utilisation et de leur importance pour l'architecture, des nécessités de réalisation et de tests
  - Exemple : classement des CU par importance

Les traitements très classiques de la gestion locale
sont à examiner en dernier (risque faible).
Les réservations (client ou gérant), mettent en jeu
une architecture plus complexe, sont à examiner en priorité :
Réservation (distante)
Réservation locale
Administration
...

Etude préliminaire

- Détailler et formaliser les cas d'utilisation
- Structurer le modèle (révision des cas si besoin)
- Faire une maquette de l'interface utilisateur
  - uniquement si l'interface est complexe ou nécessite une évaluation par le client

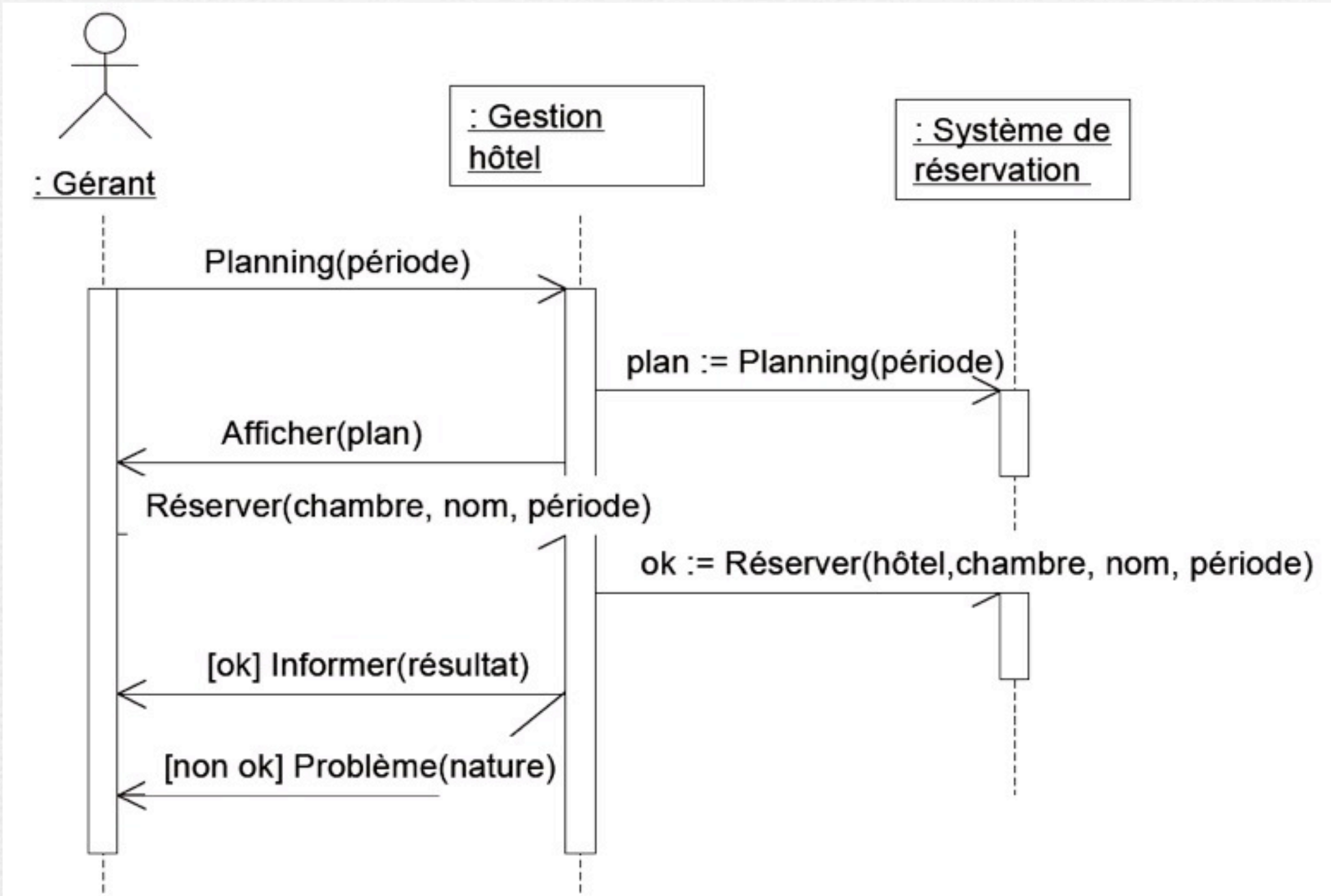
Utilisateurs non spécialistes, interface simple et logique.

Problème classique, sans risque majeur, donc pas de maquette.

Ex : Etude préliminaire

- CU : Gérer réservation par le gérant
  - **Acteur principal** : Gérant
  - **Intervenants et intérêts :** Client, Chaîne hôtelière
  - **Préconditions** : une chambre est libre pour la période désirée
  - Scénario nominal
    - 1. Le gérant demande le planning d'occupation pour la période qui vient. Le système affiche le planning sur plusieurs semaines.
    - 2. Le gérant sélectionne une chambre libre pour une date qui l'intéresse. Le système lui présente le récapitulatif de cette chambre, et sa disponibilité quelques jours avant et après la date choisie.

Ex : diagramme de séquence système pour un scénario



Etude préliminaire

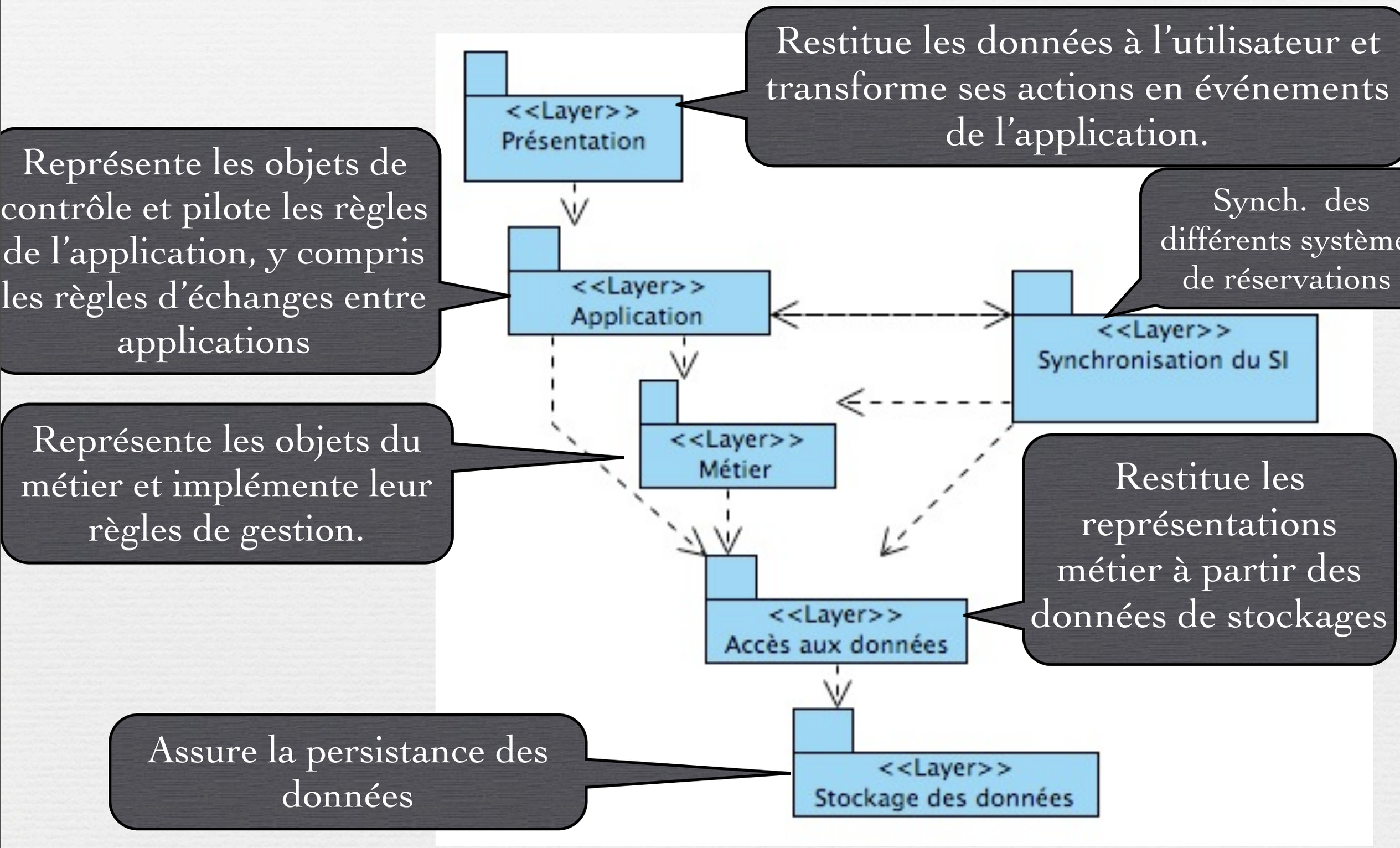
- Appréhender les besoins non fonctionnels (contraintes sur le système : environnement, plate-forme, fiabilité, vitesse…)
  - rattacher si possible les besoins aux cas d'utilisation
    - description dans les descriptions des CU (section « exigences particulières » pour UP)
    - sinon, dresser une liste des exigences supplémentaires

La chaîne possède 117 hôtels de 30 chambres en moyenne. Les appels sur le réseau sont évalués à 300 par jour (au début, prévoir des évolutions).
Pour des raisons d'extensibilité, de performances et de sécurité, la chaîne de traitement des réservations des clients doit être indépendante des liaisons des hôtels avec le système de réservation. Les hôtels ne sont pas reliés en permanence au système de réservation (économie) et les postes devront être fiables (coupures de courant...).
Le temps d'apprentissage du logiciel par les acteurs professionnels ne doit pas dépasser une demi-journée.
Une société tierce s'occupera de la maintenance du système et abritera les serveurs.

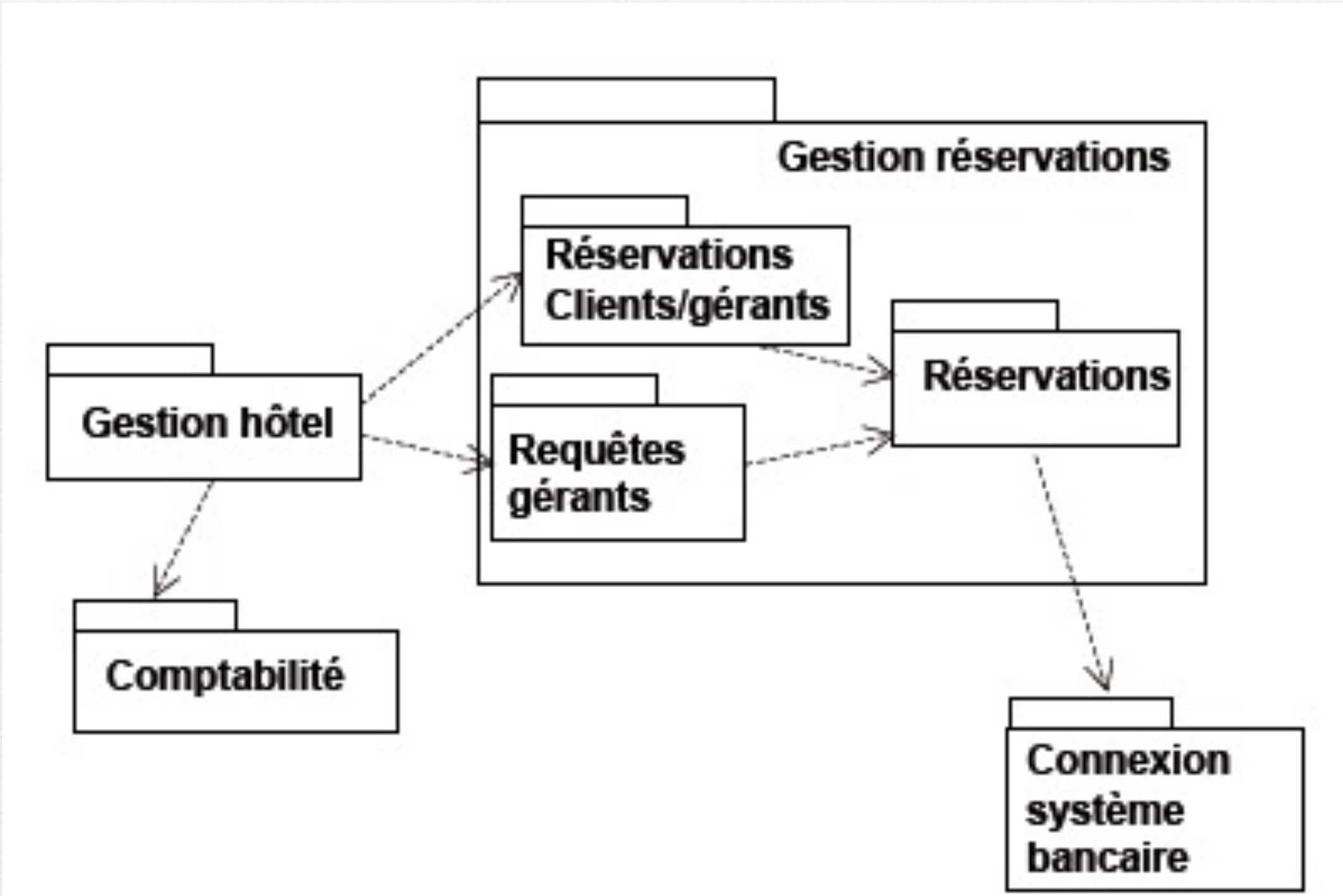
Activité : analyse

- Objectif :
  - construction du modèle d'analyse pour préparer la conception
    - forme générale stable du système, haut-niveau d'abstraction
    - vision plus précise et formelle des CU, réalisation par des objets d'analyse
  - passage du langage du client à celui du développeur
- Analyse architecturale
  - identifier les paquetages d'analyse (découpage en catégorie)
    - regroupement logique indépendant de la réalisation
    - relations de dépendances, navigabilité entre classes de paquetage différents à partir des CU et du domaine
    - point de départ du découpage en sous-systèmes

Ex : Organisation logicielle



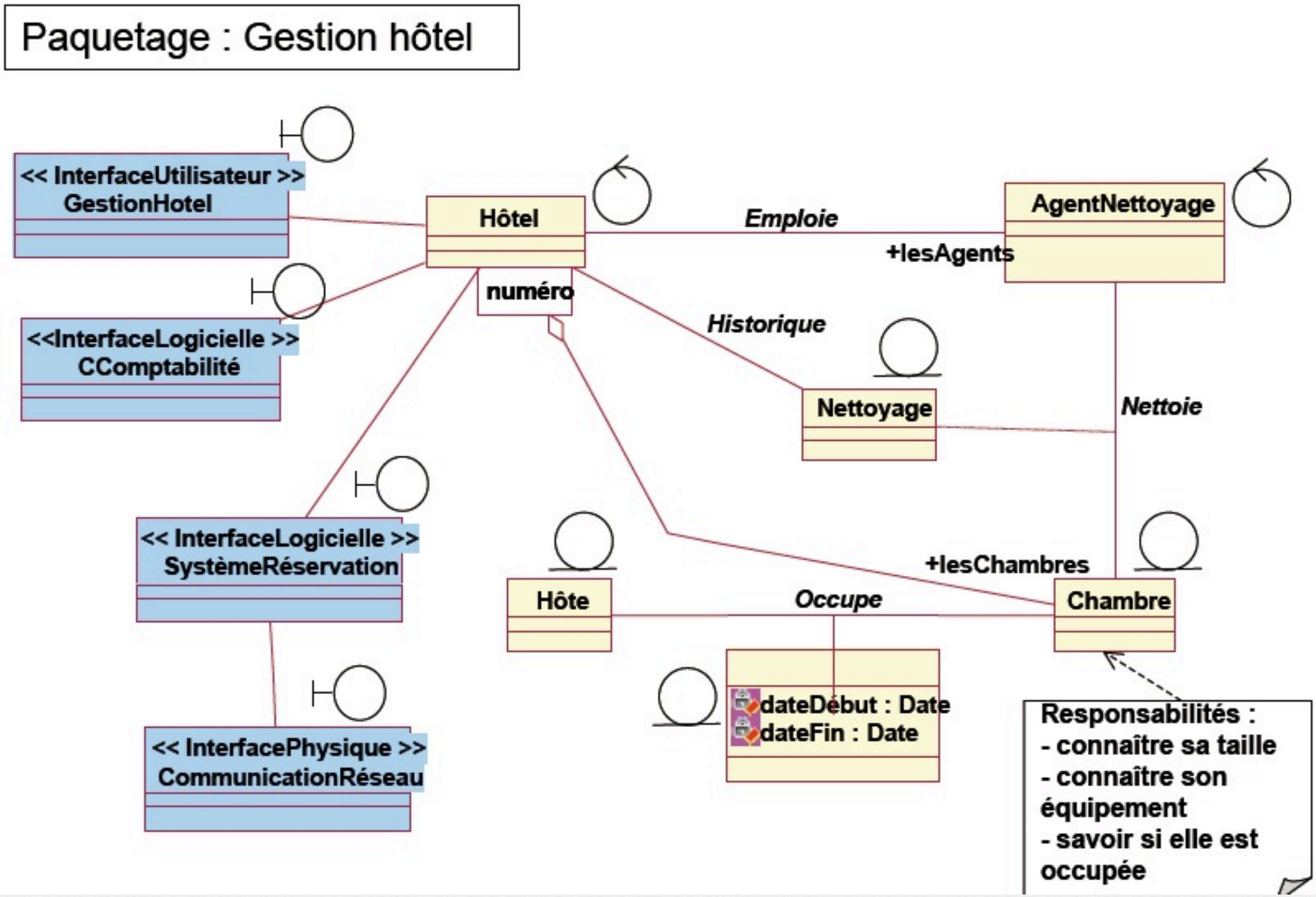
Ex : découpage en paquetages



Analyse architecturale (suite)

- Identifier les classes entités manifestes (premier modèle structurel)
  - modèle des 10-20 classes constituant l'essence du domaine
  - 3 stéréotypes de classe : frontière, contrôle, entité
  - responsabilités évidentes
- Identifier les exigences particulières communes
  - distribution, sécurité, persistance, tolérance aux fautes…
  - les rattacher aux classes et cas d'utilisation

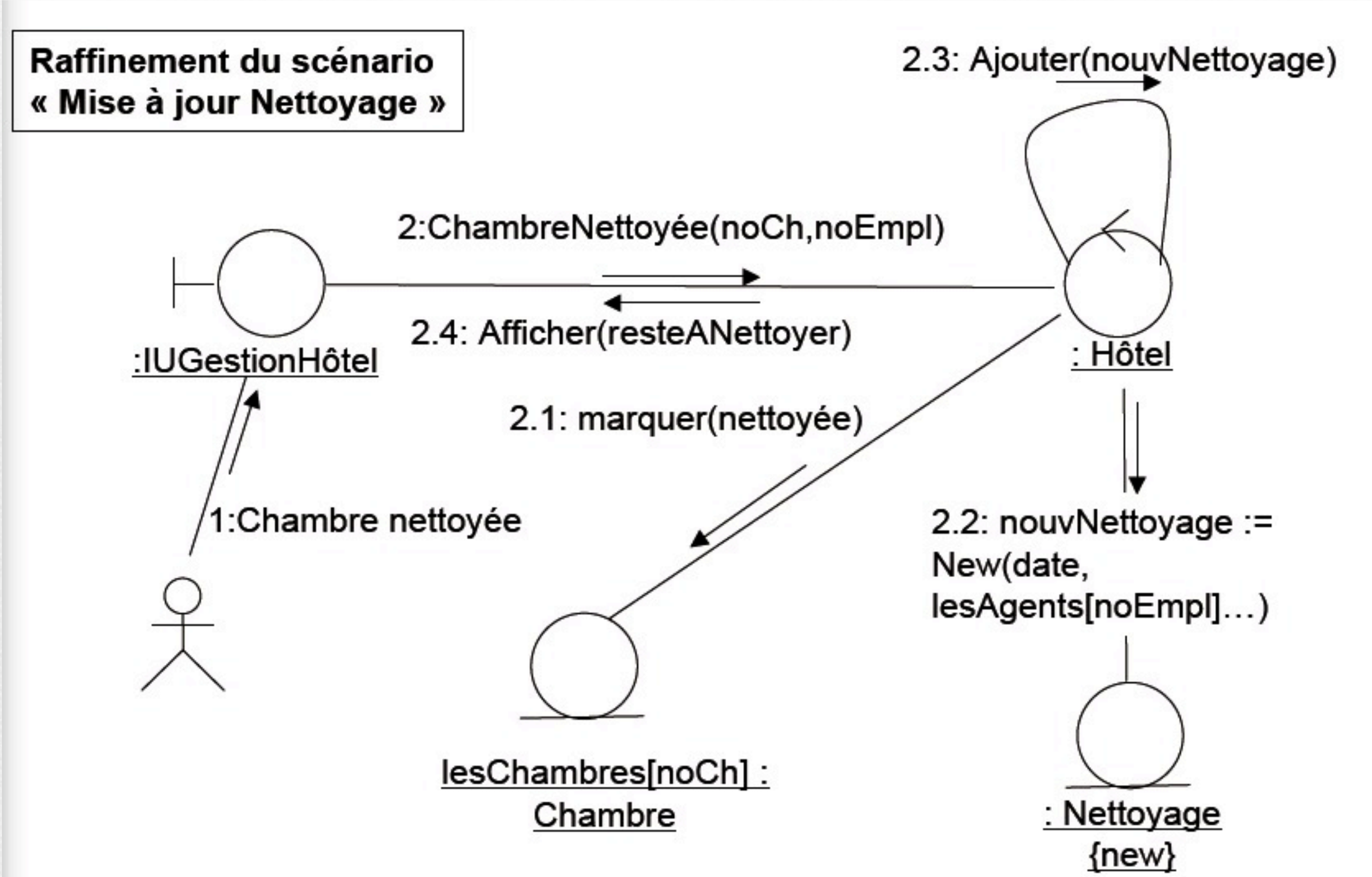
Analyse architecturale (suite)



Activité : analyse (suite)

- Analyse des cas d'utilisation
  - raffinement de tous les scénarios des cas d'utilisation →
    - découverte des classes, attributs, relations, interactions entre objets, et des besoins spéciaux
  - identifier les classes, attributs et relations
    - examiner l'information nécessaire pour réaliser chaque scénario
    - **ajouter les classes isolant le système de l'extérieur (interfaces physiques, vues externes des objets…)**
  - éliminer les classes qui n'en sont pas : redondantes, vagues, de conception, etc.

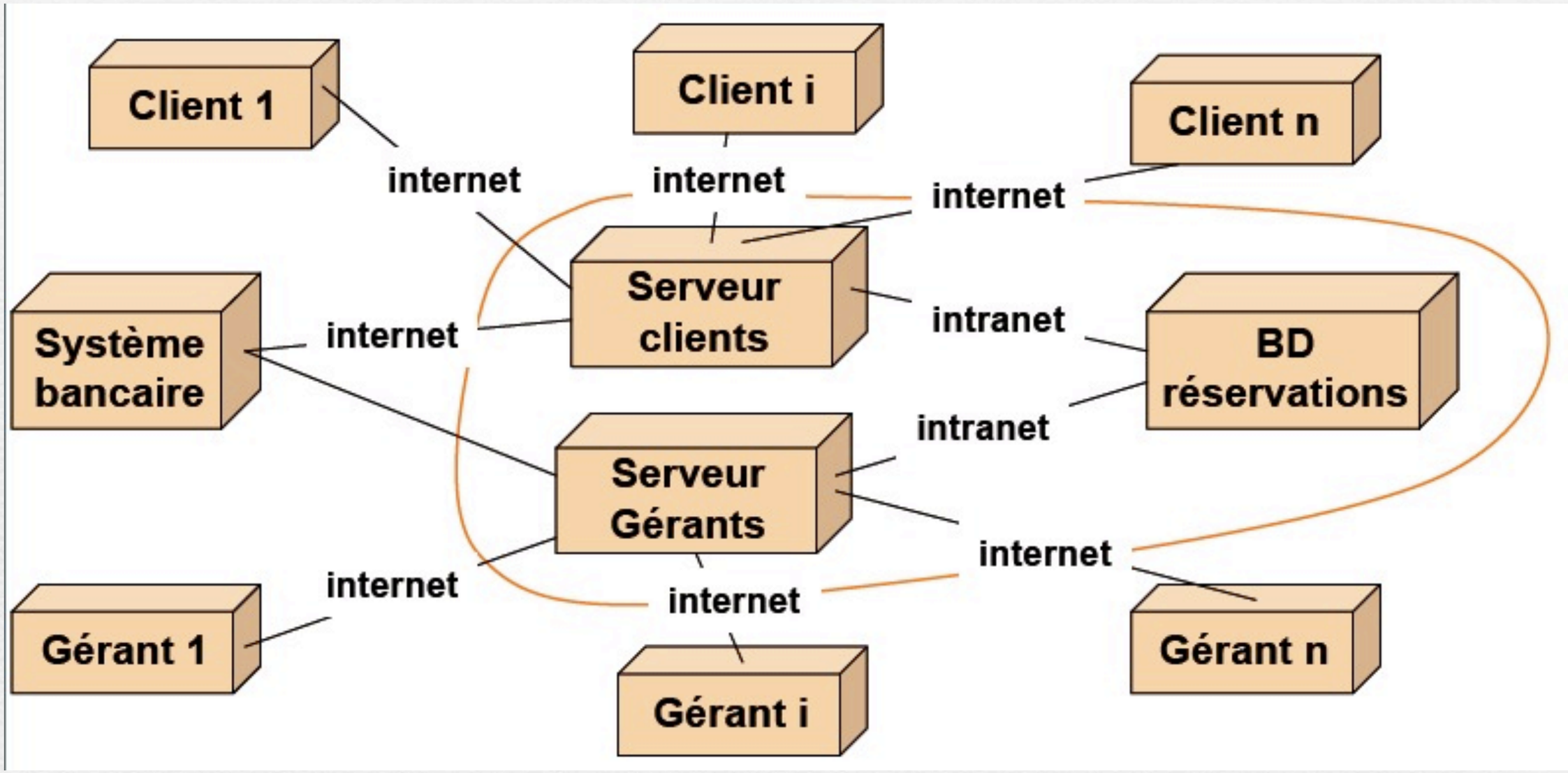
Exemple : diagramme de collaboration



3- Activité : conception

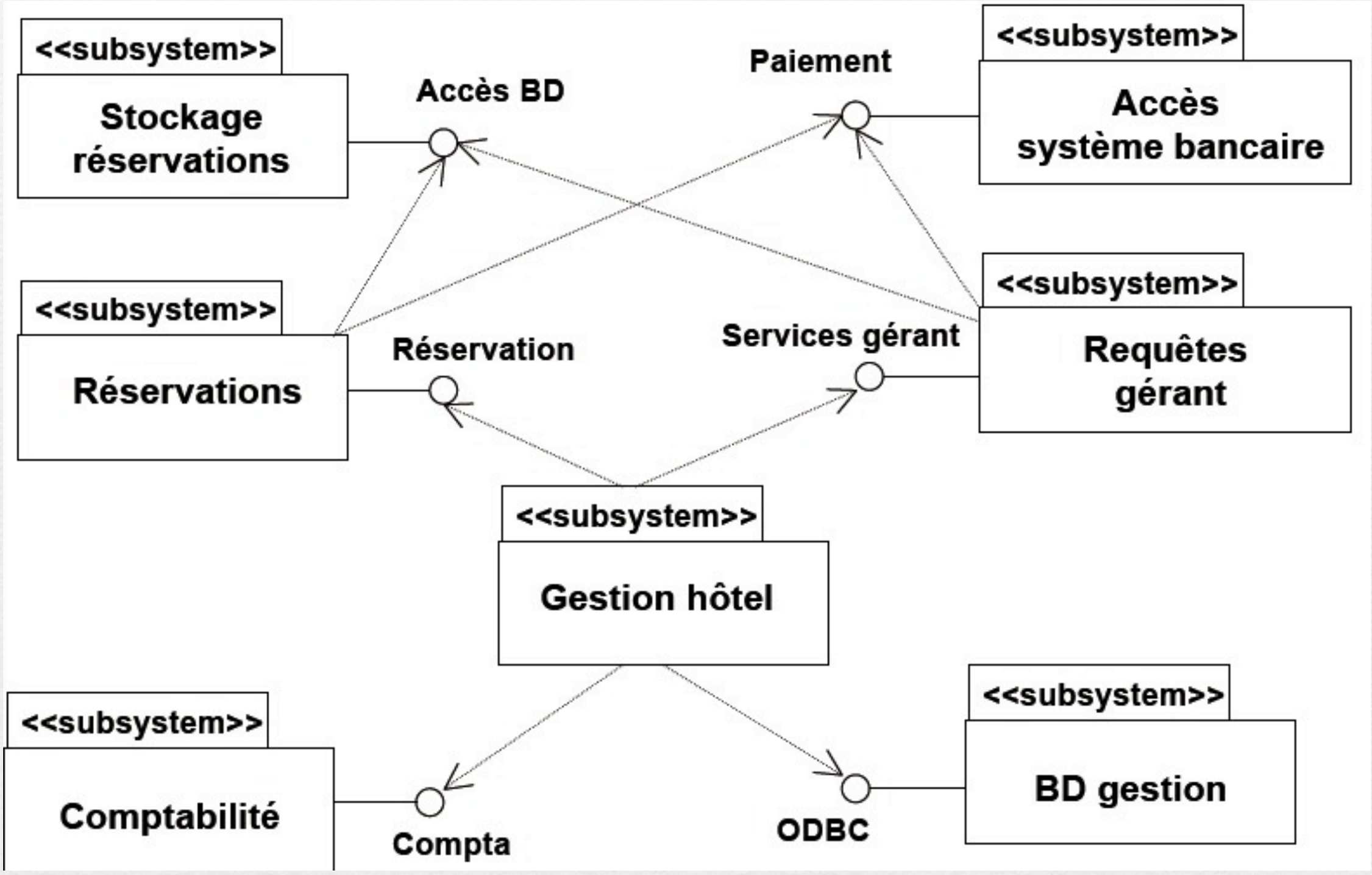
- Propose une réalisation de l'analyse et des cas d'utilisation en prenant en compte toutes les exigences
- Conception architecturale
  - identifier les noeuds et la configuration du réseau (déploiement), les sous-systèmes et leurs interfaces (modèle en couche en général), les classes significatives de l'architecture
- Concevoir les cas d'utilisation
  - identifier les classes nécessaires à la réalisation des cas ...
- Concevoir les classes et les interfaces
  - ... décrire les méthodes, les états, prendre en compte les besoins spéciaux
- Concevoir les sous-systèmes
  - mettre à jour les dépendances, les interfaces...
  - sous-systèmes de service, liés à l'appli, de middleware…
  - permettra de distribuer le travail aux développeurs

Exemple : diagramme de déploiement



- Les deux serveurs et la BD des réservations peuvent être sur un même noeud tant que les liaisons clients ne pénalisent pas les liaisons gérants
-

Exemple : découpage en sous-systèmes



4- Activité : réalisation

- Mise en oeuvre architecturale
  - identifier les artefacts logiciels et les associer à des noeuds
- Intégrer le système
  - planifier l'intégration, intégrer les incréments réalisés
- Réaliser les sous-systèmes
- Réaliser les classes
- Faire les tests unitaires
  - tests de spécification en boîte noire, de structure en boîte blanche

Bibliographie

- UNIVERSITE PARIS XII -ISIAG , MASTER 2, METHODOLOGIE ET CONDUITE DE PROJETS, CHAPITRE 3
- Génie Logiciel Orienté Objets, Philippe Collet, Master 1 Informatique, 2007-2008
- Processus de conception de SI M1 MIAGE - SIMA - 2005-2006 Yannick Prié UFR Informatique - Université Claude Bernard Lyon 1
- Méthodes de conduite de projet, Tester, optimiser, structurer ses applications Jean David Olekhnovitch, jd@olek.fr - www.olek.fr
- Les cours IBM sur le RUP
- Conduite de projet, Méthode d'analyse et de conception, Processus unifié, G. Picard, SMA/G2I/ENS Mines Saint-Etienne gauthier.picard@emse.fr, Octobre 2009
- Processus Unifié : *www2.lifl.fr/~clerbout/.../Cours4-**ProcessusUnifie**.pdf*
- *P. Colin*,